Axis Name	Considered Nodes
ancestor	Any node along the path to the root
ancestor-or-self	Same, but including the current node
attribute	Consider only attribute nodes in the tree
child	Any node directly connected to the current node
descendant	Any node from the subtree rooted at the current node
descendant-or-self	Same, but including the current node
following	Any node with id greater than the current node
following-sibling	Any same-level node with id greater than the current node
parent	The direct predecessor of the current node
preceding	Any node with id lower than the current node
preceding-sibling	Any same-level node with id lower than the current node
self	The current node

Table 1

n	a_i	b_{QHL}	s_{QHL}	t_f
n	ancestor	n	ancestors	oc=XML-Element
n	ancestor-or-self	n	{ancestors,base}	oc=XML-Element
n	attribute	n	onelevel	oc=XML-Attribute
n	child	n	onelevel	oc=XML-Element
n	descendant	n	subtree	oc=XML-Element
n	descendant-or-self	n	{subtree, base}	oc=XML-Element
n	following	root(n)	subtree	(&(oc=XMLElement)(order>order(n)))
n	following-sibling	parent(n)	onelevel	(&(oc=XMLElement)(order>order(n)))
n	parent	n	parent	oc=XMLElement
n	preceding	root(n)	subtree	(&(oc=XMLElement)(order<order(n)))
n	preceding-sibling	parent(n)	onelevel	(&(oc=XMLElement)(order<order(n)))
n	self	n	base	oc=XMLElement

Table 2

File Name	Size	Apache Cache	Overhead	HLCaches	Overhead
mondial-2.0.XML	1037629	1038094	1.00	3372502	3.25
europe-2.0.XML	317913	318384	1.00	1017080	3.20
dream.XML	149524	149982	1.00	303613	2.03
SigmodRe-cord.XML	494591	495056	1.00	1401088	2.83
books1.wml	3129	3586	1.15	8039	2.57
Average	–	–	1.03	–	2.78

Table 3

File Name	Nodes/Ops	Stor.(s)	Ops/sec.	Retr.(s)	Ops/sec.
mondial-2.0.XML	39633/57116	13.34	2970.99/4281.56	85.86	461.60/65.22
europe-2.0.XML	12783/18186	3.88	3294.59/4687.11	26.84	476.26/77.57
dream.XML	3361/6231	1.19	2824.37/5236.13	10.22	328.86/609.69
SigmodRecord.XML	15263/38518	8.43	1810.55/4569.16	56.33	270.95/683.79
books1.wml	96/138	0.0098	9795.92/14081.63	0.18	533.33/766.66
Average	-	-	2725.12/4693.50	-	384.27/659.07

Table 4

Query Patterns	Nr. Result Nodes	DOM back-end	HLCaches
/mondial/country	260	0.69	0.05
/mondial//city	3047	217.67	11.23
/mondial/country[@car_code='D']	1	6.36	2.31
/mondial//city[@is_cap='yes']	230	276.56	17.05

Table 5

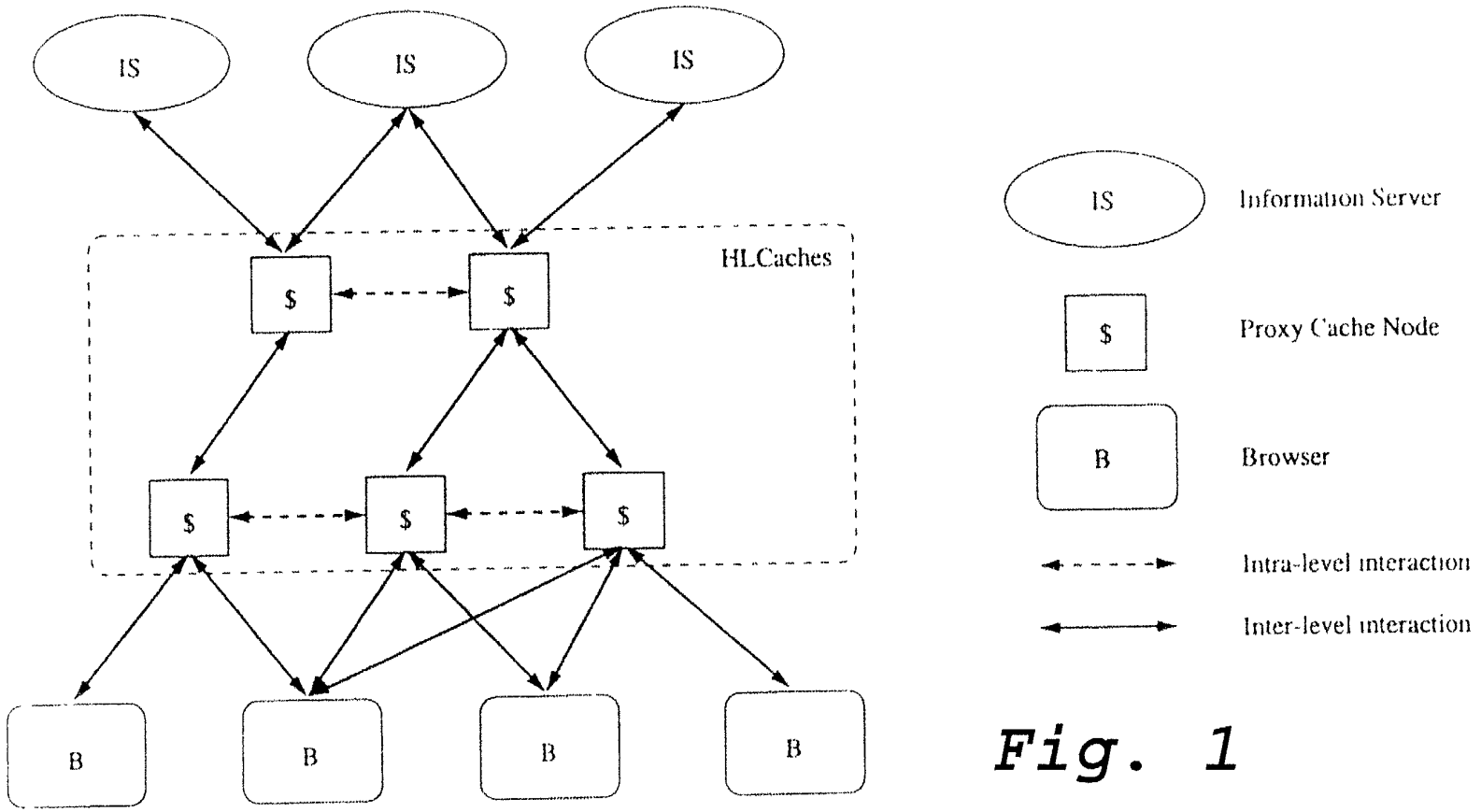
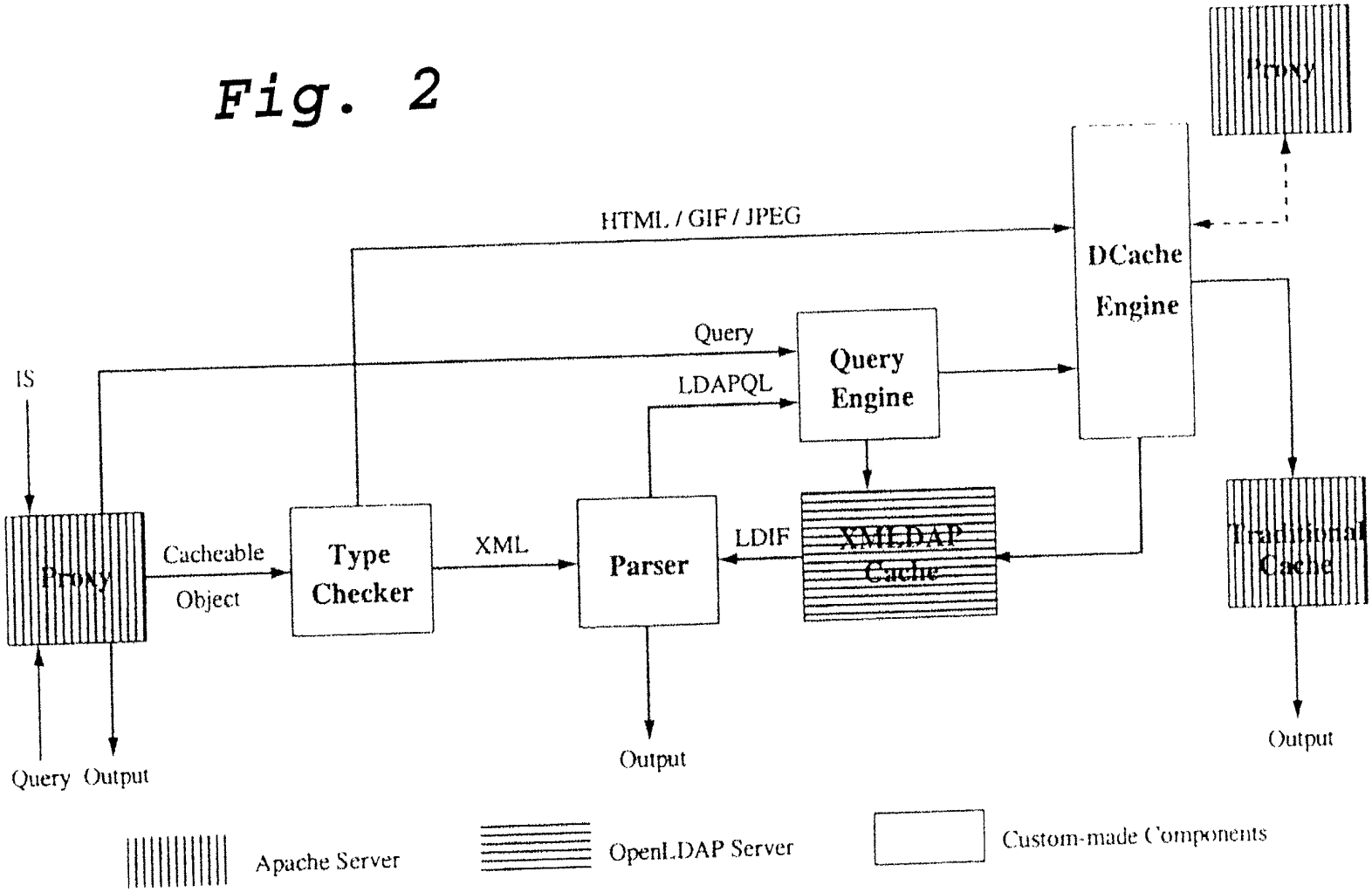


Fig. 1

Fig. 2


Proxy
HTML / GIF / JPEG
DCache Engine
Query
Query Engine
LDAPQL
IS
Proxy
Cacheable Object
Type Checker
XML
Parser
LDIF
XMLDAP Cache
Traditional Cache
Query
Output
Output
Output
Apache Server
OpenLDAP Server
Custom-made Components

```
XMLNode OBJECT-CLASS ::={
  SUBCLASS OF {top}
  MUST CONTAIN {oc,oid,name}          // required attributes
  TYPE oc OBJECT-CLASS
  TYPE oid DN                         // dns formed by oids
  TYPE name STRING
}
XMLElement OBJECT-CLASS ::= {
  SUBCLASS OF {XMLNode}
  MUST CONTAIN {order}                // required attributes
  MAY CONTAIN {value}                 // allowed attributes
  TYPE order INTEGER
  TYPE value STRING
}
XMLAttribute OBJECT-CLASS ::={
  SUBCLASS OF {XMLNode}
  MUST CONTAIN {value}                // required attributes
  TYPE value DN, STRING
}
```


Fig. 3

```
Algorithm XML2LDAP ( D )

Let  D  be an XML document to processed from left to right
/* Initialize the current node to the top of the LDAP cache
tree */
CurrentNode = "(cn=Cache,dc=top)"

while there is input  i  from  D
/* If an opening tag is found in the inventive input  i  */
     if  i  is
     <tagName attrName0=attrValue0...attrNamen=attrValuen>
              NewNode = XMLElement(tagName)
              link(CurrentNode, NewNode)
              CurrentNode = NewNode
              /* Create the attributes and link them to the
new node */
              for each attrName, attrValue pairs
                  NewAttribute = XMLAttribute(attrName,
attrValue)
                  link(NewNode, NewAttribute)

         /* If a closing tag is found in the inventive
input  i  */
         if  i  is  </tagName>
             CurrentNode = Parent(CurrentNode)

         /* else,  i  is the content of the node */
         else
             CurrentNode.value =  i
```

Fig. 4

```
<country car_code="D", area="356910",
         capital="Berlin">
  <name>Germany</name>
  <population>83536115</population>
  <languages percentage="100">
    German</languages>
  <province id="B-W", capital="cid-9",
            country="D">
    ...
  </province>
</country>
```

Fig. 5

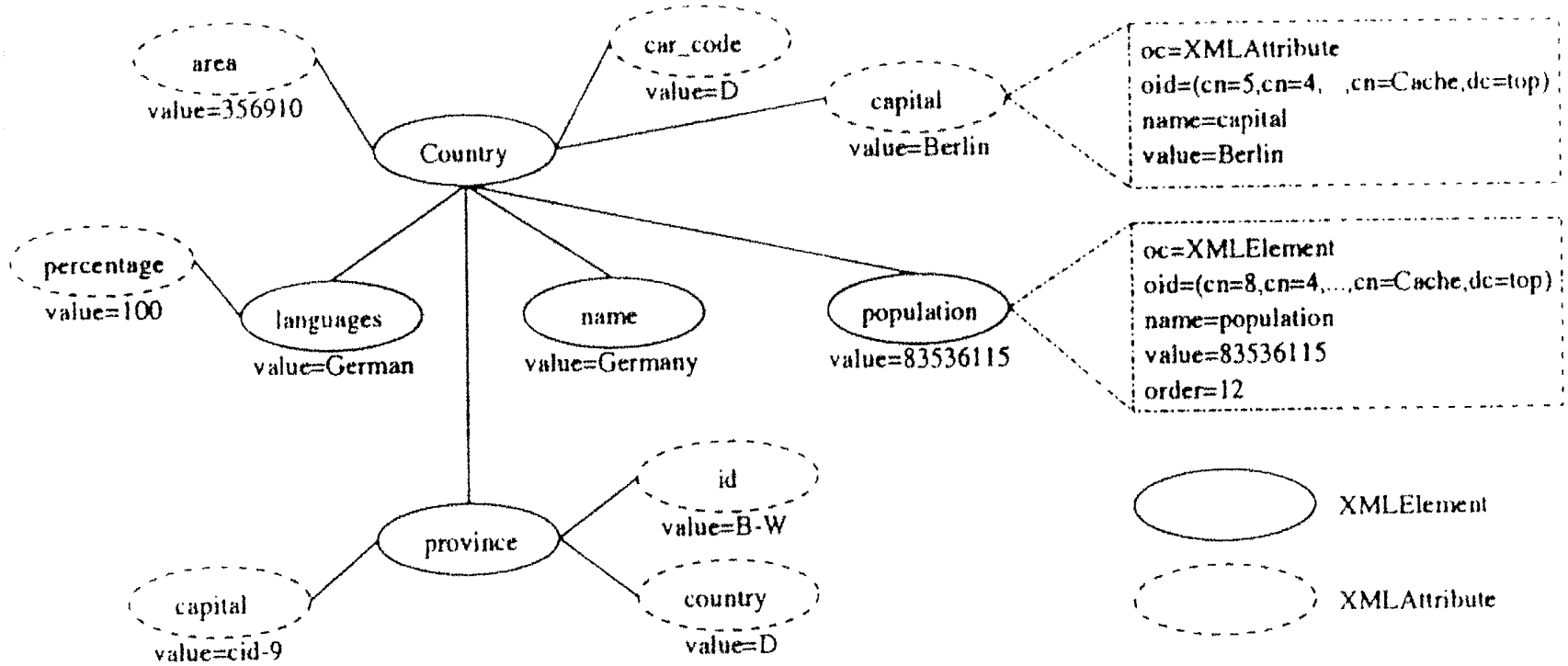

area
value=356910
car_code
value=D
capital
value=Berlin
Country
oc=XMLAttribute
oid=(cn=5,cn=4, ,cn=Cache,dc=top)
name=capital
value=Berlin
percentage
value=100
languages
value=German
name
value=Germany
population
value=83536115
oc=XMLElement
oid=(cn=8,cn=4,...,cn=Cache,dc=top)
name=population
value=83536115
order=12
id
value=B-W
province
capital
value=cid-9
country
value=D
XMLElement
XMLAttribute

Fig. 6

```
<country car_code="D" area="356910" capital="Berlin">
  <name>Germany</name>
  <population>83536115</population>
  <languages percentage="100">German</languages>
  <province id="B-W" capital="cid-9" country="D">
    <name>Baden Wurttemberg</name>
    <area>35742</area>
    <population>10272069</population>
    <city id="cid-9" is_state_cap="yes" country="D"
          province="B-W">
      <name>Stuttgart</name>
      <longitude>9.1</longitude>
      <latitude>48.7</latitude>
      <population year="95">588482</population>
    </city>
  </province>
</country>
```


Fig. 7

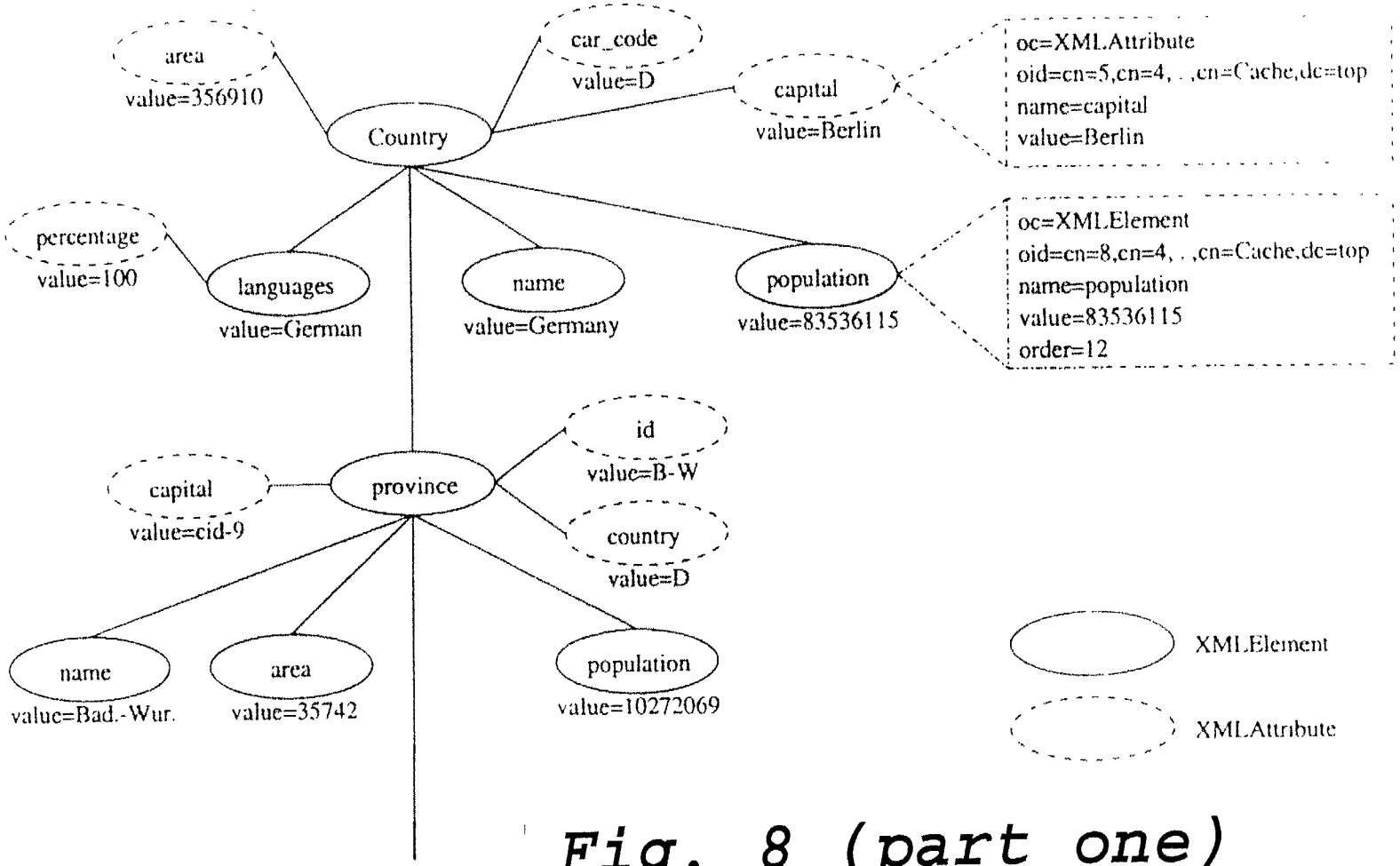


Fig. 8 (part one)

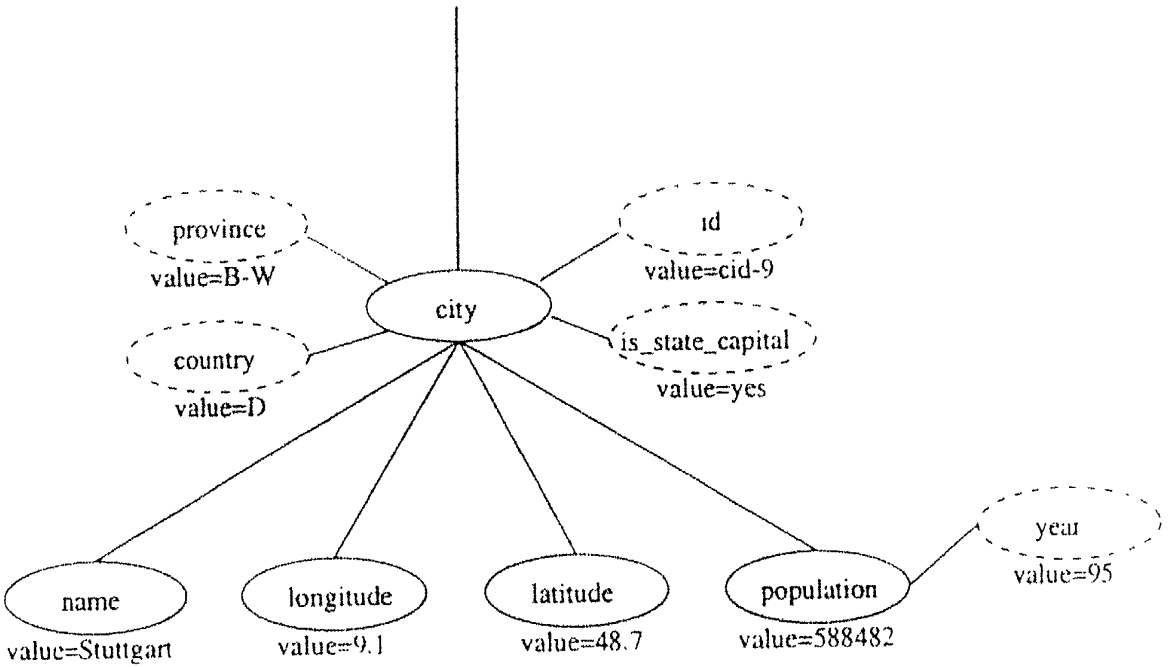


Fig. 8 (part two)

```
XMLQuery OBJECT-CLASS ::=
  SUBCLASS OF top
  MUST CONTAIN oc,hash,context,scope,xpathquery,result,
               create\_time,access\_time,popularity
  TYPE oc OBJECT-CLASS
  TYPE hash STRING
  TYPE context DN
  TYPE scope STRING
  TYPE xpathquery STRING
  TYPE result DN
  TYPE create_time STRING
  TYPE access_time STRING
  TYPE popularity INTEGER
```


Fig. 9

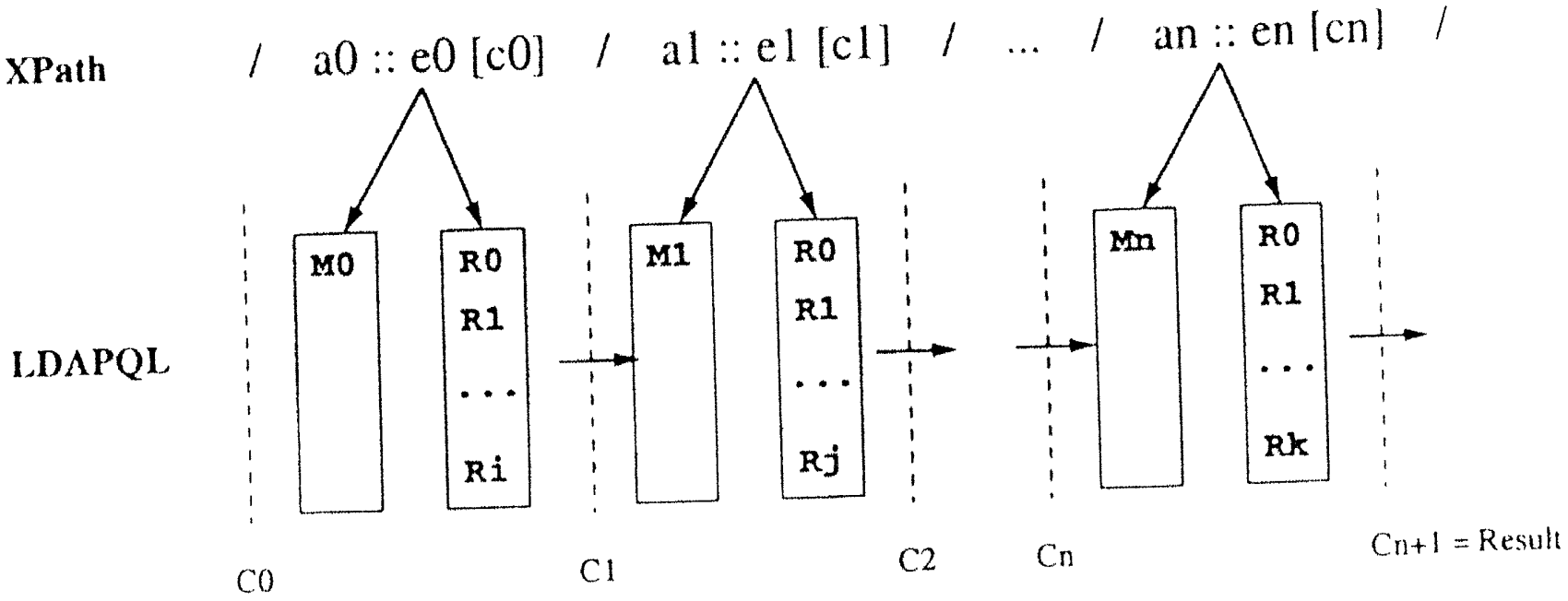


Fig. 10

```
Algorithm XPath2LDAPQL ( Q_X )

    Let  Q_X  be an XPath query
    /* Initialize  C_0  to the cache root */
     C_0 = "cn=Cache,dc=top"

    For each subquery  q_i = (C_i, w_i, C_i+1) ∈ Q_X
        /* Create a new XMLQuery node and initialize its
attributes */
        NewXMLQuery.context =  C_i
        NewXMLQuery.xpathquery =  w_i
        NewXMLQuery.hash = hash( w_i )

        /* For each node in the context, evaluate  w_i  on
it */
        for each  n ∈ C_i
             C_i+1 = C_i+1 ∈  EVAL(PET( n,w_i ))

        NewXMLQuery.result =  C_i+1
```

Fig. 11

```
Algorithm EVAL (Q, S)

    /*  Q  is an LDAPQL query (called main query) */
    /*  S = {Si}  is a set of LDAPQL queries (subordinate)
*/
    Result = LDAP( Q )
    for each subquery  Si ∈ S
        Result = Result ∩ LDAP( Si )
    Return Result

Algorithm PET( n ,  wi )

    /*  n  is a distinguished name and  wi = ai::ei[ci]  */
    Let  QHL  be an LDAPQL query (called main query)
    Let  S = {Sj}  be a set of LDAPQL queries (subordinate)

    /* Translate  ai  into  QHL = (bQHL,sQHL,fQHL,pQHL)  */
     (bQHL,sQHL,fQHL)  = BaseScope( n ,  ai )
    for each  nodeName ∈ ei
         fQHL  =  fQHL ∩ (name = nodeName)
     pQHL  =  {}

    /* Translate each predicate  cpj  into  Sj =
(bSj,sSj,fSj,pSj)  */
    Let  S  =  {}
    for each  cpj ∈ ci
        Let  cpj  be of the form  termj  opj  valuej
         (bSj,sSj,fSj)  = BaseScope(LDAP( QHL ), termj )
        for each  (nodeName,nodeValue) ∈ ci
             fSj  =  fSj ∩ (&(name = nodeName)(value =
nodeValue))
         pSj  =  {}
         S  =  S ∪ Sj

    Return ( QHL , S )
```

Fig. 12